Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Кафедра системного программирования

Лялинов Илья Михайлович

Определение частичного перекрытия кадра онлайн

Выпускная квалификационная работа бакалавра

Научный руководитель:
к. т. н. , доцент Литвинов Ю. В.

Консультант:
инженер-программист компании «Тиквижн» Осечкина М. С.

Рецензент:
исполнительный директор ООО «Системы компьютерного зрения» А. А. Пименов

Санкт-Петербург
2021

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems
Software Engineering

Ilya Lyalinov

Online detection of video frame partial overlap

Graduation Thesis

Scientific supervisor:
c. sc., assistant professor Yurii Litvinov

Scientific consultant:
software engineer at «TickVision» Maria Osechkina

Reviewer:
CEO of «Computer vision systems» Alexander Pimenov

Saint-Petersburg
2021

Оглавление

Введение

Интеллектуальные системы помощи водителю (Advanced driver-assistance systems, далее ADAS) облегчают процесс управления транспортным средством и уменьшают риск столкновения на дороге. Такие системы реагируют на некоторые изменения ситуации на дороге быстрее человека. На сегодняшний день ADAS активно развиваются, внедряются и используются в разных моделях машин. Основные функции таких систем – это автоматическая парковка, экстренное торможение, оповещение водителя о возможном столкновении, подсистема ночного видения и другие.

ADAS-системы используют разные сенсоры для сбора данных и анализа ситуации на дороге: фронтальную и боковые камеры, лидар, акселерометр, гироскоп и т. п. Для своевременного предупреждения водителя, например, о возможном столкновении необходимо обрабатывать видеопоток с фронтальной камеры в режиме реального времени. Многие потенциально хорошо подходящие для решения задач компьютерного зрения алгоритмы не могут быть применены в ADAS из-за их чрезмерной вычислительной сложности. Производительность системы важна, так как чем раньше ADAS предупредит водителя (например, о выезде на встречную полосу), тем быстрее он сможет среагировать. Чем быстрее работа ADAS, тем меньше риск дорожно-транспортного происшествия. Из-за временных ограничений разработчикам часто приходится сжимать изображения с потерей качества, использовать более дорогие и производительные микроконтроллеры, распараллеливать вычисления, использовать более быстрые, но менее точные или устойчивые модификации алгоритмов.

Установленная на машине фронтальная камера, изображение которой используется системой ADAS для анализа сцены, часто установлена так, что в ее поле зрения попадает часть капота или дворники. Иногда на лобовом стекле налипает грязь, которая перекрывает часть кадра. Это увеличивает вероятность ложно-позитивных срабатываний алгоритмов поиска объектов, сегментации и других. Определив и точно

выделив регион перекрытия, можно частично избавиться от этой проблемы, а также уменьшить актуальную для последующей обработки часть кадра. Производительность многих алгоритмов обработки видео и изображений полиномиально или экспоненциально зависит от количества обрабатываемых пикселей кадра, поэтому таким образом можно ускорить работу всей ADAS и сделать возможным применение вычислительно более сложных методов. В современных ADAS-системах регион перекрытия обычно выделяется статически перед запуском системы (далее такой способ обнаружения перекрытия будем называть статическим). Однако на лобовое стекло может налипать грязь, а камера может изменять свое положение. Понятно, что статический подход к определению перекрытия кадра не может хорошо работать в ситуациях, когда область перекрытия меняется во время работы ADAS. Автору не удалось найти решения задачи поиска частичного перекрытия видеокадра онлайн с открытым исходным кодом, то есть когда поиск перекрытия происходит во время движения автомобиля и работы системы ADAS (далее такой подход будем называть «онлайн» или обнаружением «в режиме реального времени»). Поэтому целью данной работы является создание прототипа такой системы. В дальнейшем полученное решение можно будет интегрировать в ADAS-систему компании «Тиквижн». Другие разработчики смогут использовать систему в своих ADAS проектах, а также дорабатывать и изменять ее по своему усмотрению.

1. Постановка задачи

Целью данной работы является создание модуля для обнаружения перекрытия видеокадра с фронтальной камеры автомобиля в режиме онлайн. Входными данными системы должно быть видео с фронтальной камеры автомобиля. В результате работы алгоритма на кадре должны быть определены и выделены участки перекрытия видеокадра. Для достижения цели были выделены следующие подзадачи:

- провести обзор и анализ методов, которые подходят для определения перекрытия кадра онлайн;
- сформулировать требования по качеству, времени работы алгоритма и используемым технологиям для реализации;
- реализовать модуль обнаружения перекрытия видеокадра;
- провести апробацию и тестирование системы.

2. Требования

Предполагается, что входными данными модуля обнаружения частичного перекрытия кадра является видео с фронтальной камеры автомобиля с неизвестным разрешением, типом и количеством кадров в секунду, равным 30. Результатом работы должно быть множество контуров, ограничивающих участки частичного перекрытия. Прототип системы должен точно определять закрытую дворниками, капотом или чем-то другим область кадра не дольше 2 минут с начала движения машины. Время обновления зоны перекрытия не должно быть больше 15 секунд. Одна итерация (время работы модуля между получением двух последовательных кадров) не должна длиться больше 15 миллисекунд. Разработка и тестирование проводятся на компьютере с процессором Intel Core i3-6006U 2.00GHz, использование GPU не предполагается. Перекрытие считается успешно выделенным, если удалось обнаружить не менее 90% его площади и отношение площади пересечения найденного перекрытия и достоверного (ground truth) перекрытия к площади их объединения (см. рис. (3)) $IoU \geq 0.7$.

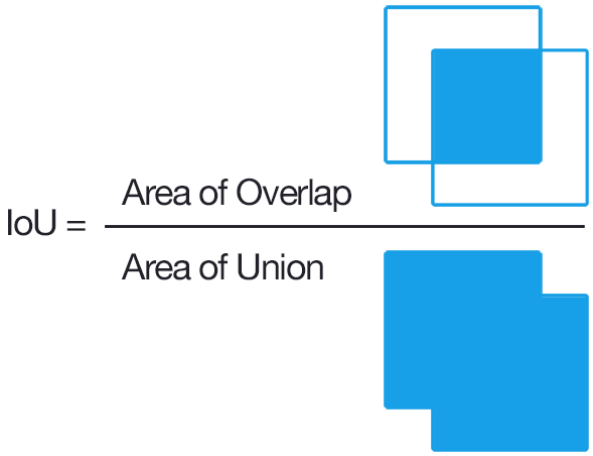


Рис. 1: Показатель IoU[1]

[1]IoU (intersection over union) for objects detection. – URL: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

3. Обзор

Перекрытие полезной части кадра на видео — это часть, которая мало меняется на протяжении всего видео. Однако статическими на видео могут оставаться и некоторые объекты сцены, не являющиеся перекрытием. Нам нужно разработать алгоритм для детекции устойчивых областей на кадре, которые перекрывают объекты сцены, минимизировав количество ложно-позитивных срабатываний. Как уже упоминалось ранее, автору не удалось найти существующих решений задачи обнаружения частичного перекрытия видеокадра в режиме реального времени. Далее рассматриваются методы, использованные для решения похожих задач, и анализируется возможность их применения в данной работе.

3.1. Обзор существующих решений

3.1.1. Обнаружение частично пересекающихся образцов

К. Артета и другие в своей статье «Learning to Detect Partially Overlapping Instances» [1] 2013 года описывают способ обнаружения на изображении объектов одного класса в случаях их высокой плотности на изображении и частичного перекрытия. Этот метод улучшает технику обнаружения объектов с помощью поиска экстремальных регионов на изображении [2]. Главная особенность алгоритма – возможность определения групп объектов. Процесс обнаружения начинается с выделения экстремальных регионов. Любые два региона либо не пересекаются, либо один полностью содержится в другом. Авторы пишут о том, что в простейшем случае такие регионы могут быть найдены поиском связных компонент на бинарном изображении. Затем каждый экстремальный регион подается на вход классификаторам, которые определяют, сколько объектов содержит данный участок. Далее с помощью процедуры вывода методом динамического программирования из всех численных оценок классификаторов и всех изначально сгенерированных регионов выбирается подмножество непересекающихся участков

(тут же определяется количество объектов на участке), на которых следующий функционал достигает максимума:

$$F(y) = \sum_{i=0}^{N} V_i(y_i) \to max, y \in Y$$

Здесь N – количество экстремальных регионов, $V_i(d)$ – численный результат классификатора региона с номером i для класса d,
$y = \{y_i | i = 1..N\}$ – переменные для оптимизации. Если $y_i = d$, то регион с номером i выбран и ему присвоен класс d. Y – множество всех y, удовлетворяющих условию $R_i \cap R_j = \emptyset => y_i * y_j = 0$

Пересекающиеся образцы детектируются как одно целое. Обучение производится с помощью машины опорных векторов. Эксперименты на реальных и синтетических изображениях показывают, что данный метод достигает точности алгоритмов, которые только считают количество объектов на изображении. Также по точности определения данный алгоритм превосходит другие методы обнаружения. Однако надо отметить и недостатки: в работе нет информации о скорости работы метода. Отсутствует информация о возможности переиспользования реализации алгоритма. Метод хуже работает при увеличении числа сгруппированных пересекающихся объектов. Это решение не может быть применимо для определения перекрытия видеокадра, так как рассчитано на обнаружение групп объектов одного класса (пешеходы, дорожные знаки и т. п.), в то время как перекрытие может быть представлено группой объектов разных классов, которые сложно предсказать заранее (капот, дворники, грязь...).

3.1.2. Обнаружение коллизий с помощью нейронной сети на основе модели нейрона саранчи

The lobula giant movement detector (далее LGMD) – детектор определения движения саранчи. Это нейрон в мозге саранчи, который наиболее сильно реагирует на изображение приближающихся объектов, например, хищника. Была разработана нейронная сеть на основе LGMD. Пре-

дупреждение столкновений под контролем этой нейронной сети было успешно не менее чем в 69% случаев. Авторы статьи 2006 года «Collision detection in complex dynamic scenes using an LGMD-Based visual neural network with feature enhancement» [3] улучшили этот алгоритм, добавив механизм усиления сгруппированных возмущений (возмущение – это набор пикселей, в которых функция интенсивности изображения принимает большие значения) и подавление изолированных для более устойчивой работы метода в условиях присутствующего шума и сложного заднего фона. Вычислительно сложные методы (например, анализа сцены или распознавания объектов) не используются, поэтому LGMD система определения возможных столкновений может работать в режиме онлайн. Минусами такого подхода является то, что алгоритм не работает при маленьких показателях контраста между приближающимся объектом и его фоном. Этот алгоритм не детектирует статическое перекрытие кадра, но может распознать появившееся перекрытие во время движения машины (налипшую грязь, например).

3.1.3. Оптический поток. Метод Лукаса-Канаде

Оптический поток [4] – это метод обнаружения движения на видео. Движущейся точке сопоставляется вектор, определяющий ее положение на следующем видеокадре. Оптический поток применяется для слежения, стабилизации видео и других задач компьютерного зрения. Условием работы алгоритма оптического потока является одинаковая яркость пикселя одного объекта на последовательных кадрах. Также у соседних пикселей должно быть примерно одинаковое движение. Метод Лукаса-Канаде [5] считает поток только для нескольких точек кадра. Уравнение оптического потока задается следующей формулой:

$$f_x u + f_y v + f_t = 0 \tag{1}$$

Где f – функция интенсивности изображения, x и y – координаты пикселя, t – время, $u = x'(t)$; $v = y'(t)$.

Этот метод достаточно быстр для слежения за точками онлайн

(для видеопотока с 30 кадрами в секунду). Однако вероятность ложноположительного срабатывания высока из-за существующих ограничений для использования алгоритма, так как в реальных условиях они не всегда выполняются.

3.1.4. Оптический поток. Метод Фарнебека

Г. Фарнебек в своей статье 2003 года «Two-Frame Motion Estimation Based on Polynomial Expansion» [6] предложил способ поиска оптического потока, основанный на решении вышеприведенного уравнения оптического потока (1). Он использовал многочлены второго порядка и весовую функцию для аппроксимации яркости окрестности пикселя. Вектор движения вычисляется для каждого пикселя изображения. Из-за этого поиск оптического потока этим способом работает дольше метода Лукаса-Канаде. Так же как и другие алгоритмы оптического потока, данный метод точно работает, только если яркость пикселей одного объекта на последовательных кадрах не меняется и у соседних пикселей примерно одинаковое движение.

3.2. Анализ существующих решений

Заметим, что нейронная сеть на основе модели нейрона саранчи была создана и используется для обнаружения приближающихся объектов, поэтому для определения статического перекрытия видеокадра этот метод не подходит. Также нет возможности применить решение из статьи «Learning to Detect Partially Overlapping Instances» [1]. Этот алгоритм может быть успешно использован для подсчета количества людей в толпе или для обнаружения и подсчета клеток на изображениях с микроскопа, но не для обнаружения перекрытия кадра на видео. Вычисление оптического потока для всего изображения можно было бы использовать для определения частичного перекрытия. Но так одна итерация алгоритма занимала бы больше 15 мс даже для видеокадров размера 160x90 пикселей (при использовании реализации алгоритма оптического потока библиотеки OpenCV). Поэтому необходимо

было самостоятельно придумать способ обнаружения частичного перекрытия видеокадра, который бы удовлетворял изложенным в главе «требования» условиям. Для обнаружения перекрытия по видеопотоку можно воспользоваться следующим соображением: если пренебречь незначительными изменениями яркости, пиксели в области перекрытия с течением времени меняться не будут. Тогда как пиксели, являющиеся проекциями дорожной сцены, будут изменяться. Заметим, что такое условие константности может быть выполнено не только для пикселей перекрытия, но и для других статических частей кадра. Для решения задачи было решено воспользоваться этим наблюдением и следующей идеей. По видеопоследовательности можно получить результирующий сглаженный или усредненный кадр и таким образом убрать на нем динамические элементы сцены и шум. Если на таком изображении четкие границы будет иметь только перекрытие, его можно будет обнаружить, применив один из алгоритмов поиска контуров на изображении. Далее рассматриваются методы поиска контуров на изображении и способы сглаживания изображения по последовательности.

3.3. Обзор используемых алгоритмов

3.3.1. Фильтр Собеля

Хороший способ выделить изменения на изображении заключается в использовании производных. Резкое изменение градиента означает существенное изменение изображения. Край объекта на картинке подразумевает скачок интенсивности пикселей. Этот скачок может быть проще рассмотрен, если мы возьмем первую производную. Таким образом поиск границ на изображении можно выполнить, находя пиксели, в которых градиент больше чем у их соседей. Оператор Собеля [7] – это оператор дискретного дифференцирования. Используя его, можно вычислить приближение градиента функции яркости изображения.

3.3.2. Фильтр Лапласа

Скачки интенсивности пикселей могут быть интерпретированы в терминах второй производной изображения. На границах объектов она обращается в ноль. Можно использовать этот критерий для обнаружения контуров на изображении. Однако нули будут образовываться не только на границах объектов, но и в других несущественных местах. Оператор Лапласа [8] для функции интенсивности изображения f определяется следующей формулой:

$$Laplacian(f) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

Результат фильтра Лапласа получается одним применением свертки с изображением. Фильтр Собеля подразумевает применение свертки изображения с вертикальным и горизонтальным ядром такого же размера, как в фильтре Лапласа. Поэтому фильтр Лапласа работает быстрее фильтра Собеля.

3.3.3. Быстрое преобразование Фурье

Дискретное преобразование Фурье [9] используется для разложения функции интенсивности изображения в ортогональном базисе из гармонических функций. Пусть дано изображение N x N и его функция интенсивности $I(x,y)$. Дискретное двумерное преобразование Фурье переводит функцию пространственных координат $I(x,y)$ в область частот (u,v):

$$F(u,v) = \frac{1}{N} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} I(n,m) e^{-2i\pi(un+vm)/N}$$

Обратное преобразование Фурье переводит частотное представление $F(u,v)$ в функцию интенсивности $I(x,y)$:

$$I(x,y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v) e^{2i\pi(un+vm)/N}$$

Преобразование Фурье возвращает набор комплексных значений. Модули возвращаемых значений содержат в себе большую часть информации о геометрической структуре изначального изображения. Но для полного восстановления изображения по его Фурье-образу нужны и модули, и аргументы получившихся комплексных значений. Высокие частоты в Фурье-образе представляют собой границы объектов и шум на изначальном изображении. Удалив все кроме них, можно избавиться от шума и сгладить границы. Вычислительная сложность быстрого преобразования Фурье для изображения N x N: $O(N * logN)$. Из минусов надо отметить, что преобразование Фурье глобальное, поэтому его использование может привести к подавлению локальных признаков изображения. Чаще всего в компьютерном зрении преобразование Фурье используется для сжатия изображений и их фильтрации.

3.3.4. Усреднение последовательности кадров

Под усреднением по n последовательным кадрам x_1, ..., x_n мы будем понимать изображение $I_n = \frac{x_1+x_2+...+x_n}{n}$.
Для пересчета среднего по последовательности фиксированной длины из последних n кадров можно было бы хранить эти последние n кадров, каждый раз вычитая из числителя «устаревший кадр» и прибавляя новый. Такой подход, однако, потребовал бы дополнительных затрат памяти, пропорциональных длине последовательности. Поэтому каждое следующее среднее изображение создается с нуля. То есть каждые n кадров мы получаем новое усредненное изображение.

3.3.5. Экспоненциальное сглаживание

Экспоненциальное сглаживание [10] часто применяется для сглаживания данных в обработке сигналов. Пусть x_t – полученный с фронтальной камеры кадр с номером t. Обозначим за s_i результат экспоненциального сглаживания на шаге i. Экспоненциальный фильтр для последовательности из набора кадров от 0 до t задается следующими формулами:

$$s_0 = x_0;$$

$$s_t = \alpha \cdot x_t + (1 - \alpha) \cdot s_{(t-1)}$$

Заметим, что такое сглаживание, в отличие от среднего, может быть вычислено после получения каждого нового кадра, а дополнительная память для хранения последовательности не требуется. Отметим также, что при маленьких параметрах α, то есть таких, что $0 < \alpha << 0.5$, вклад x_0 в результате сглаживания s_n будет максимальным до такого минимального n, при котором будет верно $(1 - \alpha)^n < \alpha$. Например, при $\alpha = 0.015$, минимальное n, удовлетворяющее описанному условию $n_{min} = 278$. Недостаток экспоненциального сглаживания применительно к нашей задаче заключается в том, что на протяжении нескольких первых кадров в результате сглаживания будут плохо размыты динамические объекты сцены, присутствующие на первом входном кадре.

4. Реализация

Инструменты

ADAS проект компании «Тиквижн» написан на языке C++. Компилируемость и возможность писать высокоуровневый код на этом языке позволяют программисту разработать эффективную и надежную систему реального времени, поэтому C++ был выбран в качестве языка реализации для этой работы. Для работы с изображениями и видео используется библиотека компьютерного зрения OpenCV.

Схема алгоритма

Основные шаги итерации модуля обнаружения перекрытия:

- получение нового кадра видео;
- сжатие входного кадра;
- вычисление размытого изображения по последовательности с полученным кадром;
- применение к среднему фильтра для определения четкой (неразмытой) области;
- объединение найденных фильтром точек в кластеры, выделение компонент;
- фильтрация ложно-позитивных компонент.

Сглаживание

Для обнаружения перекрытия было реализовано усреднение по набору последовательных видеокадров (куску из n последних кадров видео с фронтальной камеры автомобиля). Пересчет среднего происходит каждый раз после получения нового видеокадра. Для экономии памяти,

среднее по n последним кадрам создается только после получения каждого n-ого кадра. Среднее по последовательности изображение обладает следующими преимуществами: оно удаляет шум, размывает границы движущихся объектов сцены и не меняет геометрическую структуру перекрытия кадра. На тестовых видео чем больше размер последовательности, тем выше степень размытия. Такая операция позволяет найти перекрытие кадра, применив детектор контуров на изображении. Также было реализовано экспоненциальное сглаживание кадров.

Выделение четкой области на сглаженном изображении

Для последующего обнаружения перекрытия видеокадра было решено реализовать фильтр высоких частот с помощью быстрого преобразования Фурье и фильтр оператором Лапласа, чтобы потом сравнить результаты с применением этих фильтров и выбрать лучший с точки зрения скорости и качества работы. Пороговый параметр фильтра высоких частот может быть задан динамически. Если он не задан, используется значение по умолчанию. Среднее время работы фильтра Лапласа по 10 измерениям для изображения 320x180 пикселей было примерно в 9 раз меньше (среднее время работы лапласиана – 0.4 ± 0.1мс, FFT – 3.6 ± 0.1мс), чем аналогичное время работы фильтра высоких частот с помощью быстрого преобразования Фурье. Использовалась реализация методов DFT, IDFT и laplacian из библиотеки OpenCV. Тесты проводились на компьютере с процессором Intel Core-i3, графический процессор не использовался.

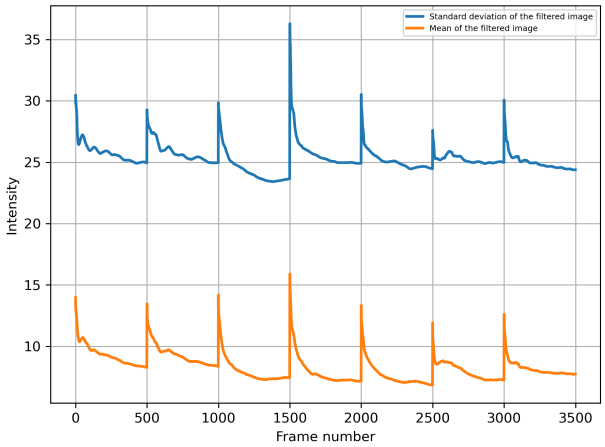
Оптимальная длина последовательности для сглаживания

Оптимальная длина последовательности для создания среднего изображения рассчитывалась исходя из нескольких условий. Во-первых, для простоты реализации было решено работать только с видеозаписями и

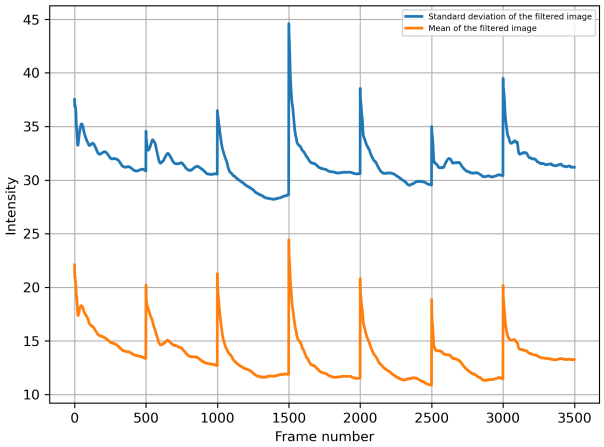
видеопотоками с частотой кадров, равной 30. Чтобы иметь возможность обрабатывать ложно-позитивные срабатывания фильтра Лапласа, было решено анализировать последовательные фильтрованные изображения. Поэтому, учитывая требование максимальной длительности работы модуля, мы установили максимальную рассматриваемую длину последовательности для создания среднего изображения n = 500. То есть на создание среднего должно уходить не более 16.67 секунды. Чтобы выбрать оптимальное количество кадров для создания среднего изображения, для видеозаписи из 3500 кадров были построены графики (см. рис. (2)) среднего и среднеквадратичного отклонения функции яркости фильтрованного усредненного. Косвенным признаком того, что мы достигли необходимой степени размытия, можно считать относительно маленькие изменения дисперсии и среднего значений функции интенсивности фильтрованного изображения при увеличении длины последовательности для создания среднего изображения. Усредненное по последовательности изображение последовательно создавалось по последним $((n-1) \bmod 500 + 1)$ кадрам, где n – номер текущего входного кадра, а фильтр применялся каждый раз после получения нового кадра. Графики строились для следующих размеров входного кадра: 640x360, 320x180, 160x90. Для отслеживания стабилизации среднего и дисперсии на интервалах $[500 * n, 500 * n + 500)$, где n – натуральное, было выбрано такое условие. Начиная с некоторой точки m на оси Ox и до конца соответствующего интервала, разница любых двух значений не должна быть больше $eps = 2.5$. Точку m, удовлетворяющую этому условию, назовем точкой стабилизации. Для каждого размера входного изображения было найдено такое минимальное k кратное 50, что для любого интервала $[500 * n, 500 * n + 500)$ точка $500 * n + k$ это точка стабилизации (см. таблицу (1)).

Кластеризация точек перекрытия

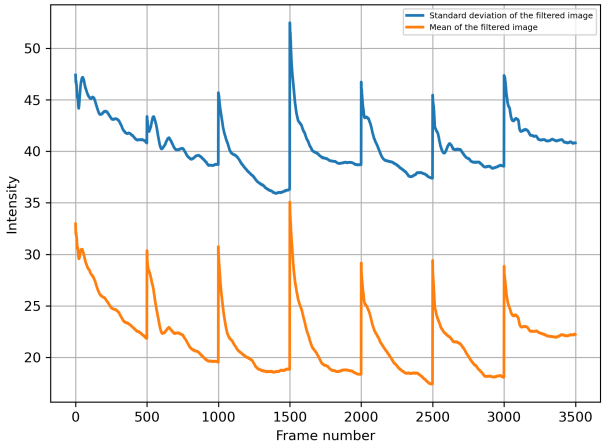
Выделять отдельные кластеры точек можно следующим образом. К результату работы лапласиана мы применяем фильтр Гаусса. Так удается



(a) 640x360



(b) 320x180



(c) 160x90

Рис. 2: графики среднеквадратичного отклонения и среднего значения функции яркости фильтрованного изображения.

Размер входного кадра	k
640x360	50
320x180	200
160x90	300

Таблица 1: оптимальная длина последовательности для различных размеров входного кадра.

уменьшить яркость одиночных найденных лапласианом точек и сгладить границы. При этом сумма яркостей всех пикселей изображения не меняется. Дальше используется взятие замыкания. Эта операция позволяет соединить близкие скопления ярких точек на фильтрованном изображении в один кластер. Затем применяется пороговый фильтр.

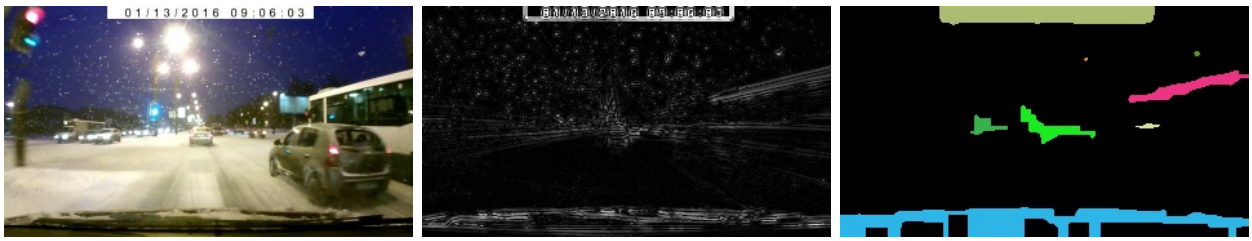


Рис. 3: слева направо: пример входного кадра; применение лапласиана к сглаженному изображению; найденные кластеры.

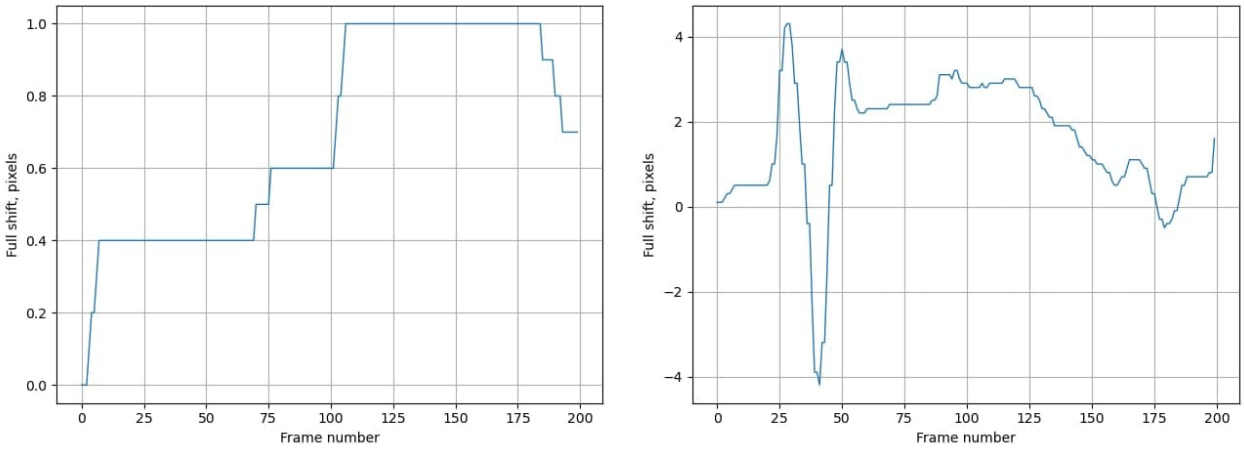
Получившиеся связные компоненты и есть искомые кластеры. Связные компоненты и их параметры (высота, ширина, площадь и другие) (рис. (3)) размечаются с помощью алгоритма BBDT (Block-based with decision trees), реализованного в библиотеке OpenCV.

Фильтрация ложно-позитивных кластеров

Каждые n кадров видео мы формируем изображение со связными компонентами. Эти компоненты представляет собой кластеры ярких точек на среднем изображении после применения детектора четкой области к сглаженному изображению. Дальше надо решить задачу отделения компонент реального перекрытия от ложно-позитивных. Ложные компоненты это части дорожной сцены, недостаточно хорошо размытые на сглаженном изображении. Для этого мы будем искать оптический поток для некоторых точек кластеров итеративным алгоритмом Лукаса-Канаде. За сдвиг компоненты за 1 кадр примем наиболее часто встречающийся сдвиг точек этой компоненты в пикселях, округленный до десятых. Так, если в кластере 9 отслеживаемых точек, три из которых сдвинулись на 1.3 пикселя, две — на 1.9 пикселя и четыре – на 0.5 пикселя, то сдвиг этого кластера будет равен 0.5 пикселя.

Со временем ошибка поиска оптического потока накапливается, поэтому раз в несколько кадров необходимо сбрасывать точки до изначального состояния и находить их сдвиг заново. Если суммарный сдвиг компоненты за все время слежения за ней больше, чем $\epsilon = 2$, мы считаем эту компоненту ложной. В противном случае рассматриваемый кластер представляет собой область частичного перекрытия кадра. Для выбо-

ра ϵ мы построили несколько графиков (рис. (4)) зависимости полного сдвига компоненты от номера кадра. Заметим, что, например, при тряске во время движения ложно-позитивная компонента будет двигаться в кадре вниз и вверх, а настоящая область перекрытия останется стабильной. Иногда движение ложно-позитивной компоненты в кадре мало и сравнимо с движением компонент реального перекрытием, тогда такой кластер «ложно» считается реальным перекрытием кадра.



(a) Реальное перекрытие

(b) Ложно-позитивный кластер

Рис. 4: графики полного сдвига компонент в зависимости от кадра.

5. Тестирование

5.1. Производительность

Все действия, необходимые для обнаружения частичного перекрытия, происходящие во временном интервале между получением двух последовательных кадров видеопотока, будем называть шагом или итерацией нашего алгоритма. Одна такая итерация может состоять из разных действий. Например, в начале работы системы, когда сглаженное изображение еще не получено, шагом алгоритма будет пересчет сглаженного изображения. Когда выделены четкие области, итерация алгоритма состоит из сглаживания и нахождения оптического потока. Также раз в несколько кадров необходимо применить детектор четкой области и выделить на полученном изображении связные компоненты. Чтобы оценить производительность одной итерации, мы измеряли время работы одной итерации пересчета сглаженного изображения, применения детектора четкой области (фильтра Лапласа), выделения связных компонент и одной итерации поиска оптического потока.

	Среднее время работы	Максимальное время работы	Среднеквадр. отклонение	Количество вызовов
Сглаживание	0.09 мс	0.24 мс	0.03 мс	1800
Детектор четкой области	0.27 мс	0.37 мс	0.04 мс	9
Кластеризация	0.8 мс	0.97 мс	0.07 мс	9
Фильтрация компонент	1.26 мс	3.89 мс	0.32 мс	1600
Итерация	1.67 мс	5.22 мс	0.43 мс	1800

Таблица 2: производительность итерации алгоритма обнаружения частичного перекрытия.

5.2. Сравнение с ground truth

Чтобы проверить, насколько точно обнаруживаются участки частичного перекрытия на видео, мы разметили области частичного перекрытия кадра на видеозаписи с фронтальной камеры машины из 2600 кадров

и вычислили показатель IoU. Мы изменяли размер входного кадра до 320x180 пикселей. Сглаженное изображение создавалось по последовательности из 200 кадров. После этого для каждого из следующих 200 входных кадров и особых точек найденных связных компонент считался оптический поток. Полученные характеристики показателя IoU представлены в таблице 3.

Среднее	Среднеквадратическое отклонение
0.73	0.14

Таблица 3: характеристика IoU.

Заключение

Были достигнуты следующие результаты:

- проведен обзор и анализ предметной области;
- сформулированы требования к реализации системы частичного перекрытия кадра;
- разработан алгоритм поиска частичного перекрытия кадра;
- реализован прототип системы обнаружения частичного перекрытия видеокадра, способный работать в режиме реального времени;
- созданный прототип протестирован на реальных видеоданных с фронтальной камеры машины при движении в городской среде.

Список литературы

[1] Carlos Arteta, Victor Lempitsky, J. Alison Noble, Andrew Zisserman. Learning to detect partially overlapping instances // In CVPR. — 2013.

[2] Carlos Arteta, Victor Lempitsky, J. Alison Noble, Andrew Zisserman. Learning to detect cells using non-overlapping extremal regions // In Proc. MIC-CAI. — 2012.

[3] Shigang Yue and F. C. Rind. Collision detection in complex dynamic scenes using an lgmd-based visual neural network with feature enhancement // IEEE Transactions on Neural Networks. — 2006.

[4] Optical flow. — https://en.wikipedia.org/wiki/Optical_flow. — Accessed on 15.12.2020.

[5] Lucas–kanade method. — https://en.wikipedia.org/wiki/Lucas%E2%80%93Kanade_method. — Accessed on 15.12.2020.

[6] Gunner Farneback. Two-frame motion estimation based on polynomial expansion // In: Image analysis,. — 2003.

[7] Sobel operator. — https://en.wikipedia.org/wiki/Sobel_operator. — Accessed on 15.12.2020.

[8] Laplace operator. — https://docs.opencv.org/3.4/d5/db5/tutorial_laplace_operator.html. — Accessed on 15.12.2020.

[9] Fast fourier transform. — https://en.wikipedia.org/wiki/Fast_Fourier_transform. — Accessed on 15.12.2020.

[10] Exponential smoothing. — https://en.wikipedia.org/wiki/Exponential_smoothing. — Accessed on 15.12.2020.